AMENDMENTS TO THE SPECIFICATION

Please amend paragraph no. [0010] of the published application (which corresponds to the paragraph found at page 2, lines 21 to 30 of the originally-filed application) as follows:

[0010] A typical scenario, which is discussed below and with reference to FIGs. 3 and 3A, begins with a user observing a page with a link to audio or video content. The link may have, for example, JavaScript that opens a new client browser window when clicked. As shown in FIG. 1, the URL that is the target of the link shown in browser 100 points to a BuildFrame Module 105 of a Media Frame Module 115 and passes, as URL arguments, a number of parameters that will be used to invoke the requested media presentation. The arguments may include, but are in no way limited to, the Broadcast Stream ID that identifies the media, the media format, the media data rate, the property, the space ID, and a series of arguments to indicate where advertisements can be played in the invoked media content. One example of URL arguments for an HTTP URL at the broadcast dot yahoo dot com domain in accordance with the present invention is:

Please amend paragraph no. [0011] of the published application (which corresponds to the paragraph found at page 1, lines 1 to 8 of the originally-filed application) as follows:

[0011] BuildFrame module 105 is invoked and receives (at step 300 of FIG. 3) a URL such as ~~by~~ the URL in the above illustrated example builds a frame set for the created client browser window 100. Continuing with the illustrated embodiment, this frame set has three parts: the top frame that is used for property identification, the player frame that is used for an embedded media player, and the data frame that is used for data presentation that is synchronized with the streaming media shown by the media player. BuildFrame module 105 parses the URL

and pulls out the property, space ID, and stream ID fields. The three enumerated fields construct a directory path name.

Please amend paragraph no. [0013] of the published application (which corresponds to the paragraph found at page 3, lines 21 to 30 of the originally-filed application) as follows:

[0013] BuildFrame module 105 (at step 305 of FIG. 3) uses an HtmlForm mechanism to invoke the frame set file when the appropriate files are found. The HtmlForm mechanism enables variable substitution with HTML files. The HtmlForm mechanism acts as a server side page evaluator. It can be used to do variable substitution as well as simple logic constructs based on the values of the variables. The HtmlForm engine is passed either as a filename or an open file pointer, a hash of variables, an output buffer, a list of handler functions and a data structure to be passed into the handler functions. In this way, the frame set can use the appropriate variables to point to the top, the player, and the data frame definitions that were identified by the BuildFrame module. The resulting web page is returned to the client browser (at step 310). The ~~These~~ individual frames may then be invoked and processed via a MediaFrame module at steps 315 and 320 of FIG. 3) described hereinafter, and the web page is returned to the client with a link to BuildList.

Please amend paragraph no. [0015] of the published application (which corresponds to the paragraph found at page 4, lines 8 to 21 of the originally-filed application) as follows:

[0015] As stated above and with reference to steps 330, 335 and 340 of FIG 3A, BuildList module 135 is called upon by media player 102 (at step 330), submits requests to an advertising server 140 (at step 335) and thereafter returns a play list of clips to play (at step 340). The returned clips may include, for example, advertisements and the selected media content. In

the illustrated embodiment, Windows Media Player ASX files are returned. However, as one skilled in the art will appreciate, any suitable media player file for the corresponding embedded media player may be returned, such as JavaScript that controls a Real Player. BuildList module 135 parses arguments contained in the calling URL and submits requests to an advertising server 140. In response, ad server 140 returns a reference to targeted streaming media advertisements. For example, the reference may be a unique ID or URL. The advertisements returned are built into the ASX play list. The play list comprises references to individual clips consisting of the advertisements and the media content. In the illustrated embodiment, the references will be to clips stored in, for example, a make play list ~~makeplaylist~~ system 145 at Yahoo! Broadcast. Media player 102 receives the play list, and for each clip in the list, connects to a media server 150 to play the clip. To illustrate with reference to FIG. 3A, media player 102 makes a call to a system, such as make play list system 145, (at step 345). The system returns a media server 150 path to the media player 102 (at step 350). Media player 102 connects to the media server 150 using the returned path and receives a media stream (at step 355).

Please amend paragraph no. [0052] of the published application (which corresponds to the paragraph found at page 15, line 32 to page 16, line 5 of the originally-filed application) as follows:

[0052] Two additional defined script commands may be embedded into every advertisement. In one embodiment, each of these script commands has a type of YAHOO! and a parameter of the form command(argument). Command identifies the command action and argument represents additional data passed to the command action. Both script commands are of the SETURL command. The SETURL command takes, as an argument, a key that is used to

index into a table to determine the URL for the DataWindow. By using this level of indirection, the actual URLs that are displayed can be changed dynamically by changing the table entries. FIG. 3B provides an example of a process flow using indirection in accordance with one or more disclosed embodiments.

Please amend paragraph no. [0053] of the published application (which corresponds to the paragraph found at page 16, lines 6 to 13 of the originally-filed application) as follows:

[0053] In one embodiment of the present invention, two SETURL commands are used. The first SETURL command is placed at 1.0 seconds into the media advertisement, and has ~~have~~ an argument of the Ad ID of the corresponding DataWindow HTML for the advertisement. For example, if the DataWindow HTML had an Ad ID of 12345, then the script command that would be inserted would have a type of YAHOO! and a parameter of SETURL(12345). When this first script command is received by the Streaming Ads System Player window (at step 360), some JavaScript will run to call the DataWindow module 125 on the server to deliver the corresponding DataWindow HTML. The DataWindow module 125 logs the viewing of the media advertisement (at step 365). The DataWindow module 125 process the DataWindow HTML, through HTMLForm, invoking a "dw" function (at step 370). The "dw" function inserts data window HTML returned by the ad server 140 into the page (at step 375). The resulting page is returned to the DataWindow frame 215.

Please amend paragraph no. [0054] of the published application (which corresponds to the paragraph found at page 16, lines 14 to 23 of the originally-filed application) as follows:

[0054] In most cases, the last SETURL command for an advertisement is the command SETURL(default). This command is used to drive the DataWindow frame 215 back

to its initial default location. Since all existing media content does not have any URL commands to drive the DataWindow frame 215, the DataWindow frame 215 for the advertisement would remain up for the entire duration of the media content, or at least until the next advertisement unless the SETURL(default) command is sent. By inserting the SETURL(default) command at 1.0 seconds before the end of the advertisement, the DataWindow frame 215 is driven back to the default initial location when media content is resumed. In one embodiment of the invention, the script command to restore the DataWindow frame 215 has a type of YAHOO! with a parameter of SETURL(default).